ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОУ ВПО НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Р.Е. АЛЕКСЕЕВА

ИНСТИТУТ РАДИОЭЛЕКТРОНИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра "Вычислительные системы и технологии"

ПРОГРАММИРОВАНИЕ

Отчёты
о выполнении лабораторного практикума

Выполнил студент группы 22-ИВТ-2 Лисенков Кирилл Алексеевич

_____«___»_____20___г.

(личная подпись) (дата)

Провел старший преподаватель кафедры «Вычислительные системы и технологии» Мартынов Дмитрий Сергеевич

_____«___»_____20___г.

(личная подпись)(дата)

НИЖНИЙ НОВГОРОД 2023

Вариант №5. Система учета. Разработать иерархию классов, которая позволит смоделировать работу системы планирования распорядка дня. Базовый класс Событие Производные классы Встреча Напомнить День рождения Заметка

```
#ifndef LIB_H
#define LIB_H

#include <iostream>
#include <string>

class Event {
protected:
    std::string name;
    std::string date;

public:
    std::string getName() const;      // Возвращает название события
    std::string getDate() const;      // Возвращает дату события
    void setName(const std::string&); // Устанавливает название события
    void setDate(const std::string&); // Устанавливает дату события
    virtual void print(std::ostream&) const; // Выводит информацию о событии в поток вывода
    virtual void read(std::istream&);        // Считывает информацию о событии из потока ввода
    friend std::ostream& operator<<(std::ostream&, const Event&); // Перегруженный оператор
вывода в поток
    friend std::istream& operator>>(std::istream&, Event&);       // Перегруженный оператор
ввода из потока
};

class Meeting : public Event {
protected:
    std::string agenda;
    std::string participants;
    std::string location;

public:
    std::string getAgenda() const;      // Возвращает повестку дня встречи
    std::string getParticipants() const; // Возвращает участников встречи
    std::string getLocation() const;    // Возвращает место проведения встречи
    void setAgenda(const std::string&);      // Устанавливает повестку дня встречи
    void setParticipants(const std::string&); // Устанавливает участников встречи
    void setLocation(const std::string&);    // Устанавливает место проведения встречи
    void print(std::ostream&) const; // Переопределение метода print для встречи
    void read(std::istream&);        // Переопределение метода read для встречи
    friend std::ostream& operator<<(std::ostream&, const Meeting&); // Перегруженный оператор
вывода в поток для встречи
    friend std::istream& operator>>(std::istream&, Meeting&);       // Перегруженный оператор
ввода из потока для встречи
};

class Alert : public Event {
protected:
    std::string text;
    std::string time;

public:
    std::string getText() const; // Возвращает текст оповещения
```

```
    std::string getTime() const; // Возвращает время оповещения
    void setText(const std::string&); // Устанавливает текст оповещения
    void setTime(const std::string&); // Устанавливает время оповещения
    void print(std::ostream&) const; // Переопределение метода print для оповещения
    void read(std::istream&);        // Переопределение метода read для оповещения
    friend std::ostream& operator<<(std::ostream&, const Alert&); // Перегруженный оператор
вывода в поток для оповещения
    friend std::istream& operator>>(std::istream&, Alert&);       // Перегруженный оператор
ввода из потока для оповещения
};

class Birthday : public Event {
protected:
    std::string birthdayBoy;
    std::string location;

public:
    std::string getBirthdayBoy() const; // Возвращает именинника
    std::string getLocation() const;    // Возвращает место проведения
    void setBirthdayBoy(const std::string&); // Устанавливает именинника
    void setLocation(const std::string&);    // Устанавливает место проведения
    void print(std::ostream&) const; // Переопределение метода print для дня рождения
    void read(std::istream&);        // Переопределение метода read для дня рождения
    friend std::ostream& operator<<(std::ostream&, const Birthday&); // Перегруженный оператор
вывода в поток для дня рождения
    friend std::istream& operator>>(std::istream&, Birthday&);       // Перегруженный оператор
ввода из потока для дня рождения
};

class Note : public Event {
protected:
    std::string text;

public:
    std::string getText() const;      // Возвращает текст заметки
    void setText(const std::string&); // Устанавливает текст заметки
    void print(std::ostream&) const;  // Переопределение метода print для заметки
    void read(std::istream&);         // Переопределение метода read для заметки
    friend std::ostream& operator<<(std::ostream&, const Note&); // Перегруженный оператор
вывода в поток для заметки
    friend std::istream& operator>>(std::istream&, Note&);       // Перегруженный оператор
ввода из потока для заметки
};

#endif // LIB_H

#include "lib.h"
#include <stdexcept>

// Class Event

std::string Event::getName() const {
    return name;
}

std::string Event::getDate() const {
```

```
    return date;
}

void Event::setName(const std::string& newName) {
    name = newName;
}

void Event::setDate(const std::string& newDate) {
    date = newDate;
}

void Event::print(std::ostream& os) const {
    os << "Event:\n"
       << "\tName: " << name << "\n"
       << "\tDate: " << date << "\n";
}

void Event::read(std::istream& is) {
    std::cout << "Name: ";
    if (!(is >> name))
        throw std::runtime_error("Failed to read!");
    std::cout << "Date: ";
    if (!(is >> date))
        throw std::runtime_error("Failed to read!");
}

std::ostream& operator<<(std::ostream& os, const Event& event) {
    event.print(os);
    return os;
}

std::istream& operator>>(std::istream& is, Event& event) {
    event.read(is);
    return is;
}

// Class Meeting

std::string Meeting::getAgenda() const {
    return agenda;
}

std::string Meeting::getParticipants() const {
    return participants;
}

std::string Meeting::getLocation() const {
    return location;
}

void Meeting::setAgenda(const std::string& newAgenda) {
    agenda = newAgenda;
}

void Meeting::setParticipants(const std::string& newParticipants) {
    participants = newParticipants;
```

```
}

void Meeting::setLocation(const std::string& newLocation) {
    location = newLocation;
}

void Meeting::print(std::ostream& os) const {
    os << "Meeting:\n"
       << "\tName: " << name << "\n"
       << "\tDate: " << date << "\n"
       << "\tAgenda: " << agenda << "\n"
       << "\tParticipants: " << participants << "\n"
       << "\tLocation: " << location << "\n";
}

void Meeting::read(std::istream& is) {
    Event::read(is);
    std::cout << "Agenda: ";
    if (!(is >> agenda))
        throw std::runtime_error("Failed to read!");
    std::cout << "Participants: ";
    if (!(is >> participants))
        throw std::runtime_error("Failed to read!");
    std::cout << "Location: ";
    if (!(is >> location))
        throw std::runtime_error("Failed to read!");
}

std::ostream& operator<<(std::ostream& os, const Meeting& meeting) {
    meeting.print(os);
    return os;
}

std::istream& operator>>(std::istream& is, Meeting& meeting) {
    meeting.read(is);
    return is;
}

// Class Alert

std::string Alert::getText() const {
    return text;
}

std::string Alert::getTime() const {
    return time;
}

void Alert::setText(const std::string& newText) {
    text = newText;
}

void Alert::setTime(const std::string& newTime) {
    time = newTime;
}
```

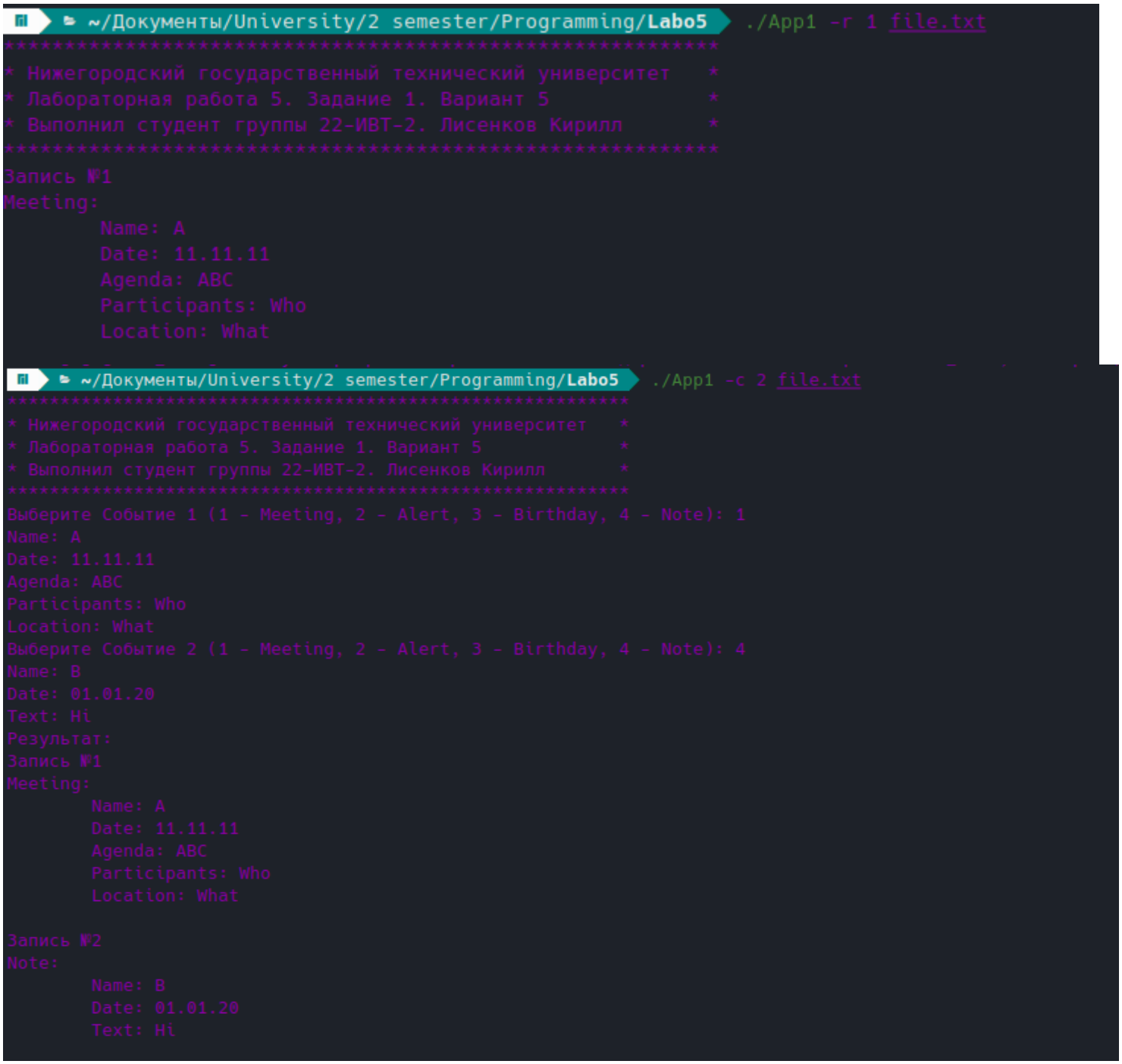
```cpp
void Alert::print(std::ostream& os) const {
    os << "Alert:\n"
       << "\tName: " << name << "\n"
       << "\tDate: " << date << "\n"
       << "\tText: " << text << "\n"
       << "\tTime: " << time << "\n";
}

void Alert::read(std::istream& is) {
    Event::read(is);
    std::cout << "Text: ";
    if (!(is >> text))
        throw std::runtime_error("Failed to read!");
    std::cout << "Time: ";
    if (!(is >> time))
        throw std::runtime_error("Failed to read!");
}

std::ostream& operator<<(std::ostream& os, const Alert& alert) {
    alert.print(os);
    return os;
}

std::istream& operator>>(std::istream& is, Alert& alert) {
    alert.read(is);
    return is;
}

// Class Birthday

std::string Birthday::getBirthdayBoy() const {
    return birthdayBoy;
}

std::string Birthday::getLocation() const {
    return location;
}

void Birthday::setBirthdayBoy(const std::string& newBirthdayBoy) {
    birthdayBoy = newBirthdayBoy;
}

void Birthday::setLocation(const std::string& newLocation) {
    location = newLocation;
}

void Birthday::print(std::ostream& os) const {
    os << "Birthday:\n"
       << "\tName: " << name << "\n"
       << "\tDate: " << date << "\n"
       << "\tBirthdayBoy: " << birthdayBoy << "\n"
       << "\tLocation: " << location << "\n";
}

void Birthday::read(std::istream& is) {
    Event::read(is);
```

```
    std::cout << "Birthday Boy: ";
    if (!(is >> birthdayBoy))
        throw std::runtime_error("Failed to read!");
    std::cout << "Location: ";
    if (!(is >> location))
        throw std::runtime_error("Failed to read!");
}

std::ostream& operator<<(std::ostream& os, const Birthday& birthday) {
    birthday.print(os);
    return os;
}

std::istream& operator>>(std::istream& is, Birthday& birthday) {
    birthday.read(is);
    return is;
}

// Class Note

void Note::setText(const std::string& newText) {
    text = newText;
}

std::string Note::getText() const {
    return text;
}

void Note::print(std::ostream& os) const {
    os << "Note:\n"
       << "\tName: " << name << "\n"
       << "\tDate: " << date << "\n"
       << "\tText: " << text << "\n";
}

void Note::read(std::istream& is) {
    Event::read(is);
    std::cout << "Text: ";
    if (!(is >> text))
        throw std::runtime_error("Failed to read!");
}

std::ostream& operator<<(std::ostream& os, const Note& note) {
    note.print(os);
    return os;
}

std::istream& operator>>(std::istream& is, Note& note) {
    note.read(is);
    return is;
}

#include "lib.h"
#include <fstream>
#include <cstring>
#include <iomanip>
```

```
int main(int argc, char** argv){
  std::cout << "*************************************************************\n" //Привествие.
              << "* Нижегородский государственный технический университет   *\n"
              << "* Лабораторная работа 5. Задание 1. Вариант 5             *\n"
              << "* Выполнил студент группы 22-ИВТ-2. Лисенков Кирилл       *\n"
              << "*************************************************************\n";
    if((argc == 2) && ((strcmp(argv[1], "-h") == 0) || (strcmp(argv[1], "--help") == 0))){
//Получения справки
        std::cout << "Справка:\n"
                  << "Добро пожаловать в программу для для хранения и обработки массива переменных структурного типа данных.\n"
                  << "  -c [N] [file_name] - запуск программы в режиме создания электронной таблицы записей, N – количество записей, file_name – имя текстового файла, в котором будет сохранен массив (таблица) записей.\n"
                  << "  -r [N] [file_name] - запуск программы в режиме чтения содержимого текстового файла file_name, на экран должны быть выведены не более N записей.\n";
        exit(0);
    }

    else if((argc == 4) && ((strcmp(argv[1],"-c") == 0) || (strcmp(argv[2],"--create") == 0))){
        // Создание таблицы записей
        if(atoi(argv[2])< 1){
            std::cout << "Ошибка, число записей не может быть отрицательным или равным нулю!";
            exit(-1);
        }
        size_t n = atoi(argv[2]);
        Event** table = new Event*[n];
            std::ofstream file(argv[3]);
        std::ofstream &rfile = file;
        for(int i{0}; i < n; i++){
            int choice;
            std::cout << "Выберите Событие " << i+1 << " (1 - Meeting, 2 - Alert, 3 - Birthday, 4 - Note): ";
            std::cin >> choice;
            switch (choice){
            case 1:
                table[i] = new Meeting();
                break;
            case 2:
                table[i] = new Alert();
                break;
            case 3:
                table[i] = new Birthday();
                break;
            case 4:
                table[i] = new Note();
                break;
            default:
                std::cout << "Некорректный выбор. Используется класс Event по умолчанию." << std::endl;
                table[i] = new Event();
                break;
            }
            std::cin >> *(table[i]);
        }
```

```
        std::cout << "Результат:\n";
        for (int i{0}; i < n; i++){
            file << "Запись №" << i+1 << "\n";
            std::cout << "Запись №" << i+1 << "\n";
            std::cout << *(table[i]);
            std::cout << "\n";
            file << *(table[i]);
            file << "\n";
        }
        for(int i{0}; i < n; i++){
            delete table[i];
        }
        delete[] table;
    }
    else if((argc == 4) && ((strcmp(argv[1],"-r") == 0) || (strcmp(argv[2],"--read") == 0))){
//Чтения бд
        if(atoi(argv[2])< 1){
            std::cout << "Ошибка, число записей не может быть отрицательным или равным
нулю!\n";
            exit(-1);
        }
        size_t n = atoi(argv[2]);
        std::ifstream file(argv[3]);
        if(!file.is_open()){
               std::cout << "Ошибка открытия файла!";
            exit(-1);
        }
        std::string line;
        int i{0};
        while (std::getline(file,line)){
            std::cout << line << std::endl;
            if(line == ""){
                i++;
            }
            if(i == n)
                break;
        }
    }
    else
        std::cout << "Неккоректные аргументы командный строки. Укажите -h или --help для
получения справки\n";
    return 0;
}
```

```
~/Документы/University/2 semester/Programming/Labo5 ./App1 -h
************************************************
* Нижегородский государственный технический университет   *
* Лабораторная работа 5. Задание 1. Вариант 5             *
* Выполнил студент группы 22-ИВТ-2. Лисенков Кирилл       *
************************************************
Справка:
Добро пожаловать в программу для для хранения и обработки массива переменных структурного типа данных.
  -c [N] [file_name] - запуск программы в режиме создания электронной таблицы записей, N - количество записей, file_name - имя текстового файла, в котором будет сохранен массив (таблица) записей.
  -r [N] [file_name] - запуск программы в режиме чтения содержимого текстового файла file_name, на экран должны быть выведены не более N записей.
```



Алгоритм "Хранение и обработка массива переменных структурного типа данных"

Начало
| Выводим приветствие
| Если аргументы командной строки равны "-h" или "--help"
| | Выводим справку по использованию программы
| | Завершаем программу
| Конец условия для получения справки
| Если аргументы командной строки равны "-c" или "--create"
| | Если второй аргумент N меньше 1
| | | Выводим сообщение об ошибке "Ошибка, число записей не может быть отрицательным или равным нулю!"
| | | Завершаем программу с кодом -1
| | Конец условия для проверки N
| | Преобразуем третий аргумент в тип size_t и присваиваем его значение переменной n
| | Создаем динамический массив указателей table размером n типа Event*
| | Открываем файл с именем, указанным в четвертом аргументе, для записи и

присваиваем его ссылку переменной file
| | Если не удалось открыть файл
| | | Выводим сообщение об ошибке "Ошибка открытия файла!"
| | | Завершаем программу с кодом -1
| | Конец условия для проверки открытия файла
| | Цикл для каждого i от 0 до n-1
| | | Вводим значение choice с клавиатуры
| | | Сравниваем значение choice с возможными вариантами
| | | | Если choice равно 1
| | | | | Присваиваем table[i] указатель на новый объект типа Meeting
| | | | Если choice равно 2
| | | | | Присваиваем table[i] указатель на новый объект типа Alert
| | | | Если choice равно 3
| | | | | Присваиваем table[i] указатель на новый объект типа Birthday
| | | | Если choice равно 4
| | | | | Присваиваем table[i] указатель на новый объект типа Note
| | | | В противном случае
| | | | | Выводим сообщение об ошибке "Некорректный выбор. Используется класс Event по умолчанию."
| | | | | Присваиваем table[i] указатель на новый объект типа Event
| | | | Конец условия выбора типа объекта
| | | Вводим значения объекта table[i] с клавиатуры
| | Конец цикла для i
| | Выводим заголовок таблицы
| | Цикл для каждого i от 0 до n-1
| | | Записываем "Запись №i+1" в файл
| | | Выводим "Запись №i+1" на экран
| | | Выводим данные из объекта table[i] на экран
| | Конец цикла для i
| | Освобождаем память, выделенную для каждого объекта table[i]
| | Освобождаем память, выделенную для массива table
| Конец условия для режима создания таблицы записей
| Если аргументы командной строки равны "-r" или "--read"
| | Если второй аргумент N меньше 1
| | | Выводим сообщение об ошибке "Ошибка, число записей не может быть отрицательным или равным нулю!"
| | | Завершаем программу с кодом -1
| | Конец условия для проверки N
| | Преобразуем третий аргумент в тип size_t и присваиваем его значение переменной n
| | Открываем файл с именем, указанным в четвертом аргументе, для чтения
| | Если не удалось открыть файл
| | | Выводим сообщение об ошибке "Ошибка открытия файла!"
| | | Завершаем программу с кодом -1
| | Конец условия для проверки открытия файла
| | Создаем строковую переменную line
| | Инициализируем переменную i значением 0
| | Цикл, пока получаем новую строку line из файла
| | | Выводим line на экран
| | | Если line пустая строка

| | | | Увеличиваем i на 1
| | | Если i равно n
| | | | Прерываем цикл
| Конец условия для режима чтения содержимого файла
| В противном случае
| | Выводим сообщение об ошибке "Некорректные аргументы командной строки.
Укажите -h или --help для получения справки"
Конец программы